CHAPTER 4

[1]

D=1 (when we transfer from memory to register).

D=0 (when we transfer from register to memory).

W=1(when we use 16 or 32 bite instructions).

W=0(when we use 8 bite instructions).

[2]

mod specifies addressing mod for selected instruction. (Have two bites).

Mod	State of R/M
00	R/M is memory(without displacement)
01	R/M is memory(byte displacement)
10	R/M is memory(2byte displacement)
11	R/M is register

[3]

w=0 so we use (8bite) of 16 bite instruction.

REG=010 so (REG is DL).

[4]

MOD=00(R/M is memory without displacement)

And we use 16 bite instructions.

Memory addressing mode is (Ds:$[BX + DI]$).

[5]

opcode	D	W
1 0 0 0 1	1	1

D=1 from R/M to REG

W=1 data word

2) 07

MOD	REG	R/M
0 0	0 0 0	1 1 1

00 R/M memory

111 DS:[Bx]

000 AX

assembly : MOV AX , DS:[BP]

[6]

1) 8B

opcode	D	W
1 0 0 0 1 0	1	1

D=1 from R/M to register

W=1 data word

2) 1E

MOD REG R/M

0 0	0 1 1	1 1 0

00 R/M is memory

011 BX

110 SS:[BP]

3) 004C

displacement

00 $(0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0)_2$

4C C4

assembly : MOV BX , SS:[BP +$(C400)_H$]

[7]

opcode	D	W
1 0 0 0 1 0	1	1

MOD	REG	R/M
0 1	1 1 0	1 1 1

2_H 0000 0010

0010 0000

machine code : $(8B772)_H$

[8]

opcode	D	W
1 0 0 0 1 0	1	1

MOD	REG	R/M
0 0	1 1 0	0 0 0

machine code : $(8B30)_H$.

[9]

The wrong is we can't move data from the register to code segment and change it because this will change the flow of the program , cs has special purpose.

[10]

All 16-bit registers:-

(AX , BX , CX , DX , SP , BP , DI , SI).

[11]

All 32-bit registers

(EAX , EBX , ECX , EDX , ESP , EBP , EDI , ESI).

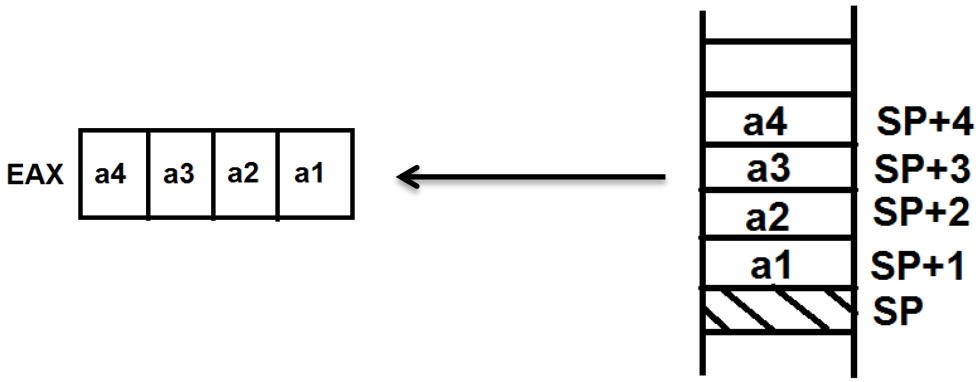
[12]

(a) PUSH AX

Copies the value which exact in the register AX to stack, we well decrease sp by one and input one byte (AH) and decrease new sp by one again and input one byte (AL).

(b) POP EAX

Remove 4 bytes from stack and put them into register EAX , we well increase sp by 4 in 4 steps , each step increase sp by 1 only and remove one byte to EAX.



[16]

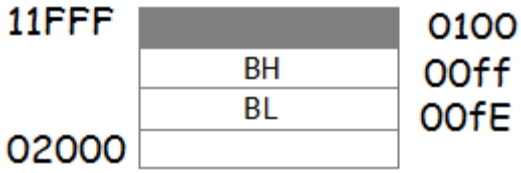
PUSHFD

[17]

SS=0200

End=(SS)0+FFFF=101FF

Sp=0100

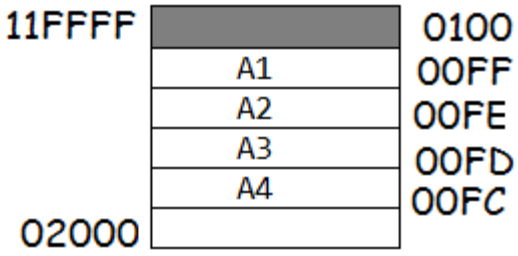


[18]

SS=0200

End=(SS)0+FFFF=101FF

Sp=0100



[19]

Loads BX from 1 word of memory location num and following 1 word is loaded to DS segment .

[20]

```
MOV BL , [NUMB]

MOV BH , [NUMB+1]

MOV DX , BX

MOV SI , BX
```

[21]

- REP use to repeat the execute of instructions .
- it is used with string instructions.

[22]

```
MOV BX , DS
MOV ES , BX
MOV SI , offset source
LEA DI ,DEST
MOV CX , 12
CLD
REP MOV  SB
```

[23]

The instruction is

```
XCHG  EBX , ESI
```

[25]

```
MOV CX,EX
MOV DS,CX
MOV AH,[BX]
```

[26]

```
XCHG  AX , BX
XCHG  ECX , EDX
XCHG  SI , DI
```

[27]

Check the flag bit E

1- if E=0 copy the content of DX into CX
2- if E=1 do nothing

[28]

Data byte :- DB

Data Word :- DW

Data Double :- DD

These directives indicate the size o the memory data addressed by the memory pointer (PTR).

[29]

```
Proc NEAR
MOV CX,04H
MOV BX,DS
MOV ES,BX
CLD
REP STOSB
```

[30]

```
PROC FAR
MOV BX,CS
MOV DS,BX
MOV AX,DATA
MOV BX,AX
```

[31]

```
MOV DX,8004H
OUT DX,BX
```

```
    INC DX
    OUT DX,BX
```

[32]

```
    MOV AL,20H
    MOV CX,256
    MOV DI,OFFSET BLOCK
    CLD
    REP STOSB
```

[33]

```
    MOV DX , A0H
    MOV CX,256
    MOV SI,OFFSET Table
    CLD
    REP OUTSB
```

[34]

```
    MOV BX,DS
    MOV ES,BX
    MOV SI,OFFSIT LIST
    LEA DI,BLK
    MOV CX,100
    CLD
    REP MOVSB
```